High-Performance Graphics (2019)
T. Foley and M. Steinberger (Editors)

Short Paper

Real-Time Ray Tracing on Head-Mounted-Displays for Advanced Visualization of Sheet Metal Stamping Defects

A. Dietrich, J. Wurster, E. Kam, and T. Gierlinger

ESI Group

Figure 1: *Left: Assessment of a physical prototype of a stamped metal sheet. Reflections of light tubes off a surface help detecting cosmetic shape defects that are difficult to quantify numerically. Right: Recreation of the light cage in a virtual reality setting, which eliminates the need for producing physical mockups.*

Abstract

Although interactive ray tracing has been around since the late 1990s, real-time frame rates had so far only been feasible for low and mid-size screen resolutions. Recent developments in GPU hardware, that specifically accelerate ray tracing, make it possible for the first time to target head-mounted displays (HMDs), which require constant high frame rates as well as high resolution images for each eye. This allows for utilizing ray tracing algorithms in novel virtual reality scenarios, which are impractical to do with rasterization. In this short paper we present our experiences of applying real-time ray tracing to the problem of detecting cosmetic defects in sheet metal stamping simulations by creating a virtual light cage.

CCS Concepts

• ***Computing methodologies*** → *Rendering; Ray tracing; Scientific visualization;* • ***Human-centered computing*** → *Virtual reality; Visualization systems and tools;*

1. Introduction

Over the last two decades, interactive ray tracing has made tremendous progress, with ray tracing solutions such as NVIDIA OptiX [PBD*10] or Intel Embree [WWB*14] having been widely available for quite some years. However, until recently these were only practical in cases where high frame rates were not a necessity (e.g., global illumination previews), whereas rasterization remained the method of choice for real-time scenarios that required consistent frame rates of 30 and more images per second. One particular area, where low frame rates are not acceptable, are virtual reality (VR) applications.

More recently, new developments have boosted ray tracing performance to a point where its use even in games becomes possible. On the hardware side, specialized ray tracing circuits have found their way into GPUs, most prominently in the form of RT cores that are part of NVIDIA's recent Turing architecture. On the software side, mainstream graphics APIs like DirectX or Vulkan have started to incorporate ray tracing extensions. These new technologies open

DOI: 10.2312/hpg.20191196

up the opportunity to exploit ray tracing algorithms also in VR to simulate optical effects that cannot easily be reproduced with rasterization based techniques (like pixel-accurate shadows and reflections). In turn, this gives way to the conception and design of novel applications of VR rendering.

One important application of VR is based in virtual prototyping, where it reduces or ideally eliminates the need for physical workflows and prototypes. In this paper, we describe a use case where ray traced reflections in immersive VR help engineers to inspect the shape of simulated stamped sheet metal (e.g., parts of a car body) in real time, allowing them to spot **cosmetic** surface defects, whose aesthetics could not be reliably estimated from numerical analysis (unlike structural defects, such a wrinkling or cracking).

Simulation of reflection lines is one of the most common methods used for virtual examination of car bodies (see, e.g., [SGA04]). Traditionally, this had been realized using reflection mapping and environment textures. With the advent of interactive ray tracing techniques, accurate real-time simulation of reflections became feasible in CAD scenarios, e.g., [WDB*06], but was limited to desktop settings with low frame rates and resolutions.

There has been prior research on bringing ray tracing to head-mounted displays, e.g., Weier et al. [WRK*16]. Most of the earlier work, however, focused on sparse frame sampling rather than full-frame ray tracing. Also, no specific industrial use case was targeted as in our case.

2. Evaluation of Cosmetic Defects in Sheet Metal Stamping

Sheet metal forming – originally introduced as early as 1880 as a replacement to forging with higher yield at 'good enough' quality – up until this day serves as one of the main production techniques in mass-market products. Today, forming as a production technique summarizes processes as diverse as stamping, punching, blanking, embossing, bending and coining. Besides sheet metal forming as the main application, processes such as stamping are in fact used with various materials including plastics. While employed by almost all production industry tiers, automotive today sees the widest use of sheet metal forming techniques.

With the recent interest into lightweight vehicles on the one hand and ever growing customer demands into visual aesthetics of exterior bodywork, the automotive industry provides some of the most interesting challenges in sheet metal stamping today. As an estimate, around half of the visual exterior of a car is made up of stamped parts. For structural integrity, an additional up to 20% of the chassis are produced from stamped parts [vTMR17]. For the context of this case study we have focused on the challenge of predicting cosmetic defects in the quality of exterior surfaces (body panels).

Besides being one of the most common processes and offering attractive benefits in cost and time taken in actual production, sheet metal forming has been notoriously difficult to master as a process. Compensating for complex effects such as springback distortion, cracking, splitting, creasing, excessive thinning or thickening of material at critical locations has long been the subject of research and efforts to simulate and predict the outcome of the manufacturing process. Beyond achieving the design goals towards structural integrity and material physics, achieving the desired aesthetic effect of a reference class A surface has been the subject of long-standing efforts in virtual prototyping [PS10]. (In automotive design, a class A surface is a set of free form surfaces having curvature and tangency alignment.) Today, simulation software such as ESI PAM-Stamp as part of ESI Sheet Metal Forming Solution provides numerical simulation results to predict most critical defects at a design stage, eliminating the need for physical prototypes [And05].

Efforts to simulate and achieve a quantifiable result to predict aesthetic quality are much more difficult to achieve, however. Simulation historically has tried to quantify the deviation to reference class A surfaces, allowing (automatic) optimization of the die face to minimize distortion introduced, e.g., by springback. However, the effect of physical non-conformity on visual aesthetics is exceedingly difficult to relate. Surface geometrical deviation as little as 25 microns has been proven to show visible effects on visual appearance [YqDC04]. While in a sub-millimeter range, precision measurements of manufactured parts in production are not able to capture these defects. In addition to the obvious issue of being too late in the process, physical measurements are not able to quantify and capture effects on visual aesthetics in practice. Deviation of shape in stamped parts relies on local curvature and deflection in contours as much as accurate coordinate matching with reference class A surfacing. Qualifying an error condition result to determine the stamped part as 'good', 'bad', or acceptable is near impossible on measurements and numerical simulation alone. Today, this process bases judgments upon prototype and pre-production parts in the try-out phase of the automotive body-in-white. In addition to being too late in the process to affect meaningful change, qualification and acceptance is based purely on expertise and experience, thus non-reproducible.

For the use case at hand, we explored two methods of analyzing cosmetic quality on real world prototypes. Stoning uses an abrasive stone of predetermined size, requiring an engineer to rub the flat face of the stone against the formed part in predefined sequences specific to the geometric intent of the component. The resulting scratch marks highlight localized changes in curvature - as the stone bridges the differing curvatures, roughing only adjacent sheet metal (Figure 2). Digital stoning tries to simulate the process and serves well to determine large-scale defects - but fails to offer a reproducible way to detect subtle broadly distributed changes in surface deflection angles.

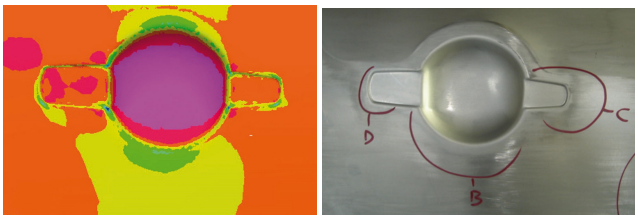

Figure 2: *Methods for assessing surface defects. Left: Numerical simulation result as a contour plot showing normal distortion. Right: Result of stoning on a physical prototype panel, where scratch marks from a stone highlight localized changes in curvature. Structural defects are easily spotted, but it is difficult to say whether any distortion of the shape is aesthetically acceptable.*

As an alternative approach, production quality is evaluated based on physical prototype reflections, shining the panel with oil and evaluating reflective behavior using an array of fluorescent lights. Based on interactive handling of the part and varying viewpoints relative to light sources, an engineer or skilled tradesperson would be able to observe if reflections are consistent across the surface (Figure 1). Any diverging lines would indicate a noticeable defect, possibly affecting aesthetics of the final panel. While difficult to reproduce and subjective, this method captures all properties of the surface relevant to visual anomalies of the final assembly.

The key aspects of modeling the real world test of panel cosmetics inside an immersive virtual reality environment focusses on interactivity and accuracy of reflections. Prior image-based solutions using reflection mapping or resolution-reduced ray tracing such as vertex tracing [UBBS01] failed to meet accuracy requirements of surface-based reflection. While brute force ray tracing solved the problem, interactive review of reflections was only possible using pre-recorded movement. While this presents a workable and repeatable solution to evaluate previously known defects - where a continuous review of engineering changes to address the problem might be required - the discovery and qualification of new issues is only possible using an exploratory, interactive process as per the real world procedure.

Our work presents an approach to address the interactive evaluation of likely cosmetic defects in immersive virtual reality as an application of real-time ray tracing (Figure 3). Analytical ray traced reflections provide the required accuracy and interactivity to allow engineers to preload the qualification of manufacturing effects on as-designed class A surfacing. Results are available continuously from initial design to final manufacturing, allowing virtual prototyping of the full chain of production steps from stamping to fitting to treatment to paint. Reproducible results allow demonstration, discussion and interactive reviews of the cosmetic effect of small-scale predicted defects in stamping simulation results.

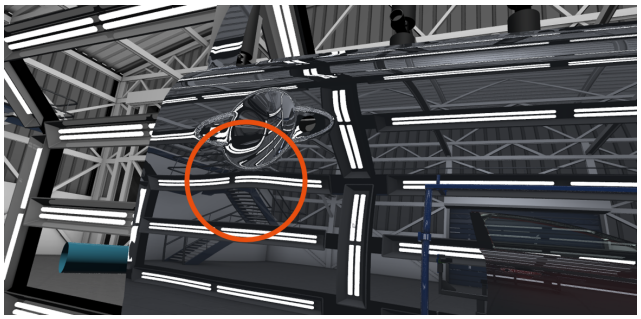

Figure 3: *Analysis of ray traced reflection behavior on a simulated prototype in immersive virtual reality. The part marked by the red circle exhibits a smooth curvature, but still may be aesthetically unacceptable.*

3. Ray Tracing on HMDs

Implementation of our VR ray tracing system has been realized as an extension to our in-house rendering engine, which is used in a variety of different virtual prototyping solutions. The engine features a hierarchical structure that encapsulates external rendering APIs in dynamically loaded backend modules. In our case, we are using OpenGL and NVIDA OptiX as external frameworks.

3.1. Rendergraph

The rendering engine is capable of traversing a rendergraph which controls the backends. Figure 4 illustrates how the graph looks for an HMD ray tracing scenario. It can be seen that there are not only 3D rendering passes (CLEAR_GL, SCENE_GL, SCENE_RT, WIDGET), but also several fullscreen and post-processing passes like color grading or filtering. The main reason for employing a graph that contains both rasterization and ray tracing passes is to enable hybrid rendering, i.e., to only render specific parts of an image with ray tracing or rasterization. However, it is important to note that for the use case presented in this paper, the scene is always fully ray traced, while the OpenGL passes only generate depth buffers and 3D widgets. This is because the current implementation only supports hybrid rendering on a per-pixel basis, i.e., pixels can either be rasterized or ray traced. Since viewers will typically stand close to a reflective surface, ray tracing would have to be used for almost all pixels anyway.

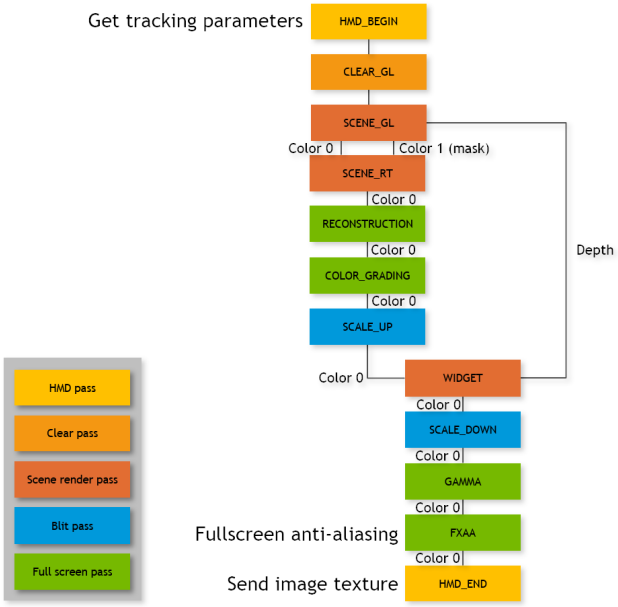

Figure 4: *Rendergraph for HMD ray tracing. The graph defines all passes involved in generating a frame. Most important parts are the scene rendering passes using OpenGL (SCENE_GL) and OptiX (SCENE_RT). Other passes include 3D widget generation (WIDGET) and fullscreen post-processing (green). Blit passes (blue) are used to adapt different target sizes so that widgets can be rendered at a higher resolution. Yellow stages perform no actual rendering tasks, but instead communicate with the HMD. They have been embedded into the rendering pipeline in order to minimize latency.*

A very important part of the HMD rendergraph are the stages responsible for communication with the display. These are actually no rendering passes, but are implemented as stages in our rendering pipeline. The reason for doing this is to minimize latency as much

as possible, which is crucial for the immersive effect created by an HMD. This way, the viewing parameters of the headset are queried as close as possible to image computation. Likewise the final image is sent right after generation has finished.

3.2. Vision-Matched Rendering

Targeting an HMD as display provides a number of opportunities for performance optimization, as due to the characteristics of the human visual system not all parts of an image have to be computed with the same accuracy.

Hidden area masking. To accommodate for the lens distortion caused by the HMD optics, a rectangular image needs to be warped before it can be displayed. This causes parts of the image to be compressed, some parts, specifically at the borders, are not visible at all. The shape and size of the hidden area can vary per HMD. OpenVR provides SDK functions to request a so-called *hidden area mesh*. Typically, the hidden area mesh is rendered into the stencil buffer, which prevents fragment shaders from being invoked for invisible pixels. In a ray tracing engine this is even easier to exploit by simply masking generation of primary rays.

In all our experiments we used an HTC Vive Pro HMD (see Section 4). For the Vive Pro, the hidden area mesh is almost circular (Figure 5). We therefore do not need an explicit mask buffer, but rather skip pixels in the OptiX ray generation program, in case they lie outside a disk inscribed in the image square. Furthermore, experiments showed that the radius of the masking circle does not have to be of maximal extend. Depending on the distance of the HMD lenses to the display, a radius of 80%-90% of the maximum was possible without being noticeable. With an 80% setting, the surface area of the disk is roughly half the area of the full image square. The frame rate increases accordingly.

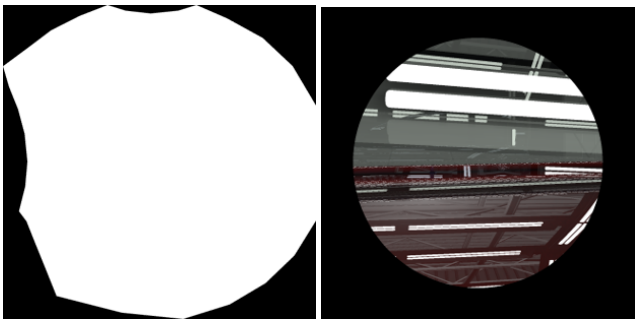

Figure 5: *Left: Hidden image areas due to lens distortion. Right: Approximation of the visible area with a disk. The disk radius selected is 80% of the maximum radius (which is half the box width).*

Variable rate sampling. A particular problem with HMD ray tracing is aliasing. Due to constant head motion, aliasing causes strong flickering, which in our case is specifically visible in the reflection of light tubes (Figure 6). An obvious approach to reducing aliasing in a ray tracer is super sampling. Since full oversampling would degrade our frame rate, we experimented with selective oversampling strategies: Even though the Vive Pro does not perform eye tracking, we can assume that a user will typically look to the center of the display. We therefore increase the sampling rate only near the center. Figure 6 shows an example: Here an area with a radius of about half the size of the display disk radius was chosen and four samples per pixel taken (red). In addition, dithering (i.e., randomized switching of the sampling rate) was used to avoid a sharp border where the sampling rate changes. Unfortunately, while this looks promising on a desktop display, on an HMD flickering of light reflections was not mitigated much. To have a noticeable effect, sampling rates above 16 samples per pixels were needed, which, however, did significantly reduce the frame rate, even with a very small oversampling area. Therefore, we did not use variable rate sampling in the final setup (see Section 4).

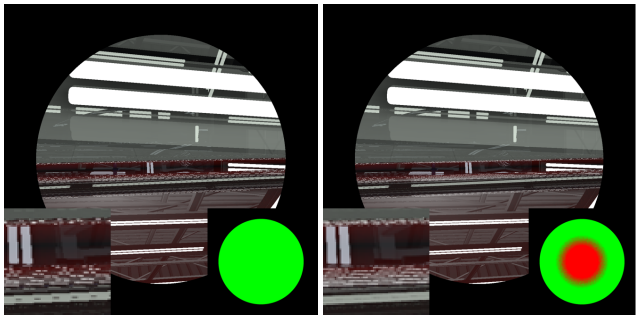

Figure 6: *Variable rate sampling. Green area: 1 sample per pixel, red area: 4 samples per pixel. Oversampling is dithered randomly to avoid a sharp border between the two areas.*

3.3. Fullscreen Anti-Aliasing

Another anti-aliasing method that is regularly used in highly dynamic rendering applications, most notably gaming, is to perform a post-process filtering step. An efficient algorithm is *Fast ApproXimate Anti-Aliasing* (FXAA) [JGY*11]. In our rendering pipeline (Figure 4) this is implemented as an OpenGL stage, which can be executed without a noticeable performance impact. FXAA is an edge-aware low-pass filter that detects edges based on contrast difference. For pixels that are found to be near an edge, an approximate luminance gradient is computed. Samples are then taken along the axis perpendicular to the gradient, which approximately relates to the orientation of the edge. Finally, the resulting pixel color is computed as a weighted average of the samples.

The effect of FXAA can be seen in Figure 7. While it does a good job at smoothing jagged edges of directly visible light tubes (left magnification), it doesn't work as well on noise-type aliasing of reflections (right magnification). Rather than fusing the small fragments of reflected light tubes, it smooths the borders around the fragments.

4. Performance Results

As hardware configuration for our prototype system we used an NVIDIA Quadro RTX 8000 GPU in combination with an HTC Vive Pro HMD. The RTX 8000 is fitted with 4608 CUDA cores, 576 Tensor cores, and 72 RT (ray tracing) cores. OptiX makes use

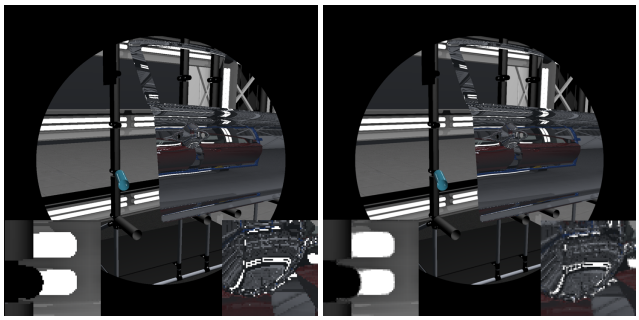

Figure 7: *Left: FXAA disabled. Right: FXAA enabled. Inlays show a magnification of light tubes and their reflections.*

of RT cores for hardware accelerated traversal and intersection operations, whereas other operations, such as shading, are executed on CUDA cores. The Vive Pro features a display resolution of 1140×1600 pixels per eye at a 90 Hz refresh rate. It has a 110 degree field of view.

The 3D geometry data used in our experiments consists of 2.2 million triangles, collected into about 1000 individual objects. The scene contains data exported from the sheet metal stamping simulation plus a small amount of geometry for the light cage and background. We used one point light and up to 3 levels of reflection, which amounts to at most 8 rays per sample (1 primary ray, 3 reflection rays, 4 shadow rays). In addition, parts of the scene contain precomputed ambient occlusion lighting (which does not affect performance). Shaders assigned to the sheet metal parts are purely reflective. Additionally, the scene includes some car door models covered with a red car paint material that consists of both diffuse and specular components (see Figure 1).

As as a target resolution for the ray tracing backend the default HMD setting of 2016×2240 pixels per eye was used. This provides basic oversampling for every pixel, which was necessary to avoid excessive flicker. Edges were further smoothed by an FXAA fullscreen pass (see Section 3.3). As mentioned in Section 3.2, additional super sampling at the image center was not beneficial and therefore omitted.

With this setup, the ray tracing backend is able to produce about 20-45 fps depending on the user's view point (20 fps when the complete view shows reflections). It is important to note that frame rates of the ray tracer and the HMD display are decoupled. In order to reduce judder caused by latency, the HMD driver performs an *asynchronous reprojection* [Vla16], where older frames are warped to match the actual camera position. In our case, for positional camera changes a slight judder is visible. However, based on comparison with an OpenGL rendered test scene, we found that the remaining judder is quite subtle and acceptable.

5. Conclusion

In this paper, we presented an approach for the interactive evaluation of potential cosmetic sheet metal stamping defects in an immersive virtual reality environment. The application of real-time ray tracing allows engineers to evaluate sheet metal stamping simulations without the need for building physical prototypes or reliance on loosely correlated numerical parameters. This demonstrates that on current GPUs, full-frame ray tracing is possible today even on HMDs, which require high resolutions and frame rates.

Our system is still early work in progress. A particular problem is posed by noise-type aliasing that is visible in the reflections of area lights. To this end, we plan to experiment with more advanced adaptive super sampling methods based on foveated rendering techniques and eye tracking.

References

[And05] ANDERSSON A.: Evaluation and visualization of surface defects – a numerical and experimental study on sheet-metal parts. In *Numerical Simulation of 3D Sheet Metal Forming Process* (Aug. 2005), Smith L., Zhang L., Wang C.-T., Shi M. F., Yoon J.-W., Stoughton T. B., Pourboghrat F., (Eds.), vol. 778 of *American Institute of Physics Conference Series*, pp. 113–118. 2

[JGY*11] JIMENEZ J., GUTIERREZ D., YANG J., RESHETOV A., DEMOREUILLE P., BERGHOFF T., PERTHUIS C., YU H., MCGUIRE M., LOTTES T., MALAN H., PERSSON E., ANDREEV D., SOUSA T.: Filtering approaches for real-time anti-aliasing. In *ACM SIGGRAPH Courses* (2011). 4

[PBD*10] PARKER S. G., BIGLER J., DIETRICH A., FRIEDRICH H., HOBEROCK J., LUEBKE D., MCALLISTER D., MCGUIRE M., MORLEY K., ROBISON A., STICH M.: Optix: A general purpose ray tracing engine. *ACM Trans. Graph. 29*, 4 (July 2010), 66:1–66:13. 1

[PS10] PARK D., SREEJITH P. S.: Analysis of surface deflection in autobody outer panel. *World Academy of Science, Engineering and Technology 61* (01 2010), 5–9. 2

[SGA04] SUSSNER G., GREINER G., AUGUSTINIACK S.: Interactive examination of surface quality on car bodies. *Computer-Aided Design 36* (04 2004), 425–436. 2

[UBBS01] ULLMANN T., BRUDERLIN B., BEIER D., SCHMIDT A.: Adaptive progressive vertex tracing in distributed environments. In *Proceedings Ninth Pacific Conference on Computer Graphics and Applications. Pacific Graphics 2001* (Oct 2001), pp. 285–294. 3

[Vla16] VLACHOS A.: Advanced vr rendering performance. In *Game Developers Conference 2016* (2016). 5

[vTMR17] VON THADEN G., MOGGE F., RIEDERLE S.: Roland berger global market study: Automotive metal components for car bodies and chassis. URL: https://www.rolandberger.com/publications/publication_pdf/roland_berger_global_automotive_stamping_study_e_20170210.pdf. 2

[WDB*06] WALD I., DIETRICH A., BENTHIN C., EFREMOV A., DAHMEN T., GUNTHER J., HAVRAN V., SEIDEL H., SLUSALLEK P.: Applying ray tracing for virtual reality and industrial design. In *2006 IEEE Symposium on Interactive Ray Tracing* (Sep. 2006), pp. 177–185. 2

[WRK*16] WEIER M., ROTH T., KRUIJFF E., HINKENJANN A., PÉRARD-GAYOT A., SLUSALLEK P., LI Y.: Foveated Real-Time Ray Tracing for Head-Mounted Displays. *Computer Graphics Forum* (2016). 2

[WWB*14] WALD I., WOOP S., BENTHIN C., JOHNSON G. S., ERNST M.: Embree: A kernel framework for efficient cpu ray tracing. *ACM Trans. Graph. 33*, 4 (July 2014), 143:1–143:8. 1

[YqDC04] YANG L. X., QING DU C. Y., CHENG F.: Recent development for surface distortion measurement. In *16th WCNDT 2004 – World Conference on non-destructive testing* (2004). 2